4 AGILE MYTHS

As with any framework or method, myths and misunderstandings can gain credence and become ‘common knowledge’ over time. This chapter discusses common myths that many people associate with Agile and explains why they are just myths and not facts.

MYTH: AGILE IS NEW

Agile is certainly not new. Agile methods have been around for a long time. The frameworks that collectively are now known as ‘Agile’ mainly evolved in the late 1980s and 1990s, which means that Agile is mature and an approach that is inherently familiar to most people. In essence Agile is all about enabling inspection and adaptation in dynamic environments where variability is experienced. This is a founding principle of complexity theory, chaos theory and evolutionary theory. It is also the way human beings interact with the world on a day-to-day basis – indeed it is the only way human beings can interact effectively with a complicated and complex world.

MYTH: IMPLEMENTING AGILE IS EASY

This common misunderstanding is the single biggest challenge when endeavouring to implement Agile delivery capabilities in an organisation or team. While Agile frameworks are inherently simple, the effort needed to implement an effective Agile operating model (see Section 5.1) is not as it requires cultural change. It is usually not easy to change a complex systems delivery lifecycle (SDLC) to a simple one; organisations normally find it easier to make things more complex than to simplify them.

Sadly, what happens in some organisations is that they try to implement an Agile operating model or a single Agile framework ‘by the book’ and without understanding the transformational complexity. Therefore they either fail to implement Agile, or they do achieve some success but at significantly higher cost and pain than they would have done if they had managed the transformation more effectively. Such organisations inevitably fail to achieve the true benefits of Agile. You can theoretically learn to fly a plane by reading a book, but don’t expect me to sit next to you on your first take-off!

MYTH: AGILE GIVES INSTANT BENEFIT

While a transformation to Agile can deliver huge benefits, the reality is that the majority of transformations go through a ‘learning curve’. Whilst people and organisations are learning, the delivery capability can actually go downwards before it makes the step-change upwards and begins to achieve the new improved delivery capability.

MYTH: AGILE MEANS NO DOCUMENTATION

This myth most likely stems from a misinterpretation of the Agile Manifesto where

it states ‘We value working software over comprehensive documentation’. It is important to understand that the Manifesto does **not** say that documentation is not required, it says that the focus is on producing working software instead of spending exhaustive amounts of time creating detailed documentation up front.

All effective Agile deliveries should allow for and produce focused, value-driven, business-beneficial documentation that enables the business to use the product effectively and the technical team to support and maintain it. Failing to produce appropriate documentation would be a classic example of technical debt (see Section 10.4.1).

MYTH: AGILE MEANS ‘HACKING’ CODE TOGETHER WITH LITTLE THOUGHT OR DESIGN

‘Hacking’ in Agile means ‘cobbling together’ an IT system with little or any design or architectural thinking.

The Agile Manifesto (see Section 1.2) states that ‘Continuous attention to technical excellence and good design enhances agility’ and many Agile frameworks provide the tools and techniques for the team to produce very high quality code. For example, in eXtreme Programming (XP – see Section 14.1), many practices are aimed specifically at ensuring that the quality of the product being delivered is fit for purpose. This includes the practice of ‘Pair Programming’, where developers work in pairs to help each other write the best designed software assets that can be created (see Section 14.1.4). This provides constant quality assurance (QA) and leads to continuous refactoring (see Section 8.10.1), a technique used to increase robustness and simplification of the design of the system without changing the behaviour of the system.

Many Agile teams subscribe to the concepts of ‘Software Craftsmanship’ (McBreen, 2002). A software craftsman creates easy-to-understand software that is designed well, documented well and is easy to build on in the future. In this context it is also important to understand that, in an Agile development, one of the main results should be the delivery of functionally and technically fit-for-purpose products. This means that it is perfectly acceptable for an Agile team to reduce the quality level for a tactical product that does not require high quality.

MYTH: AGILE IS A ‘SILVER BULLET’

Agile is not the answer to all IT problems. There is no single answer to all IT problems; rather it’s about integrating different frameworks that each provides part of the answer. The implementation of delivery and management frameworks such as Agile must be pragmatic. It must recognise the real-world environment in which a system will be implemented and used, and consider the best integration of Agile and non-Agile frameworks that will work in that real-world environment – there is no single ‘silver bullet’ framework.

This means that, once a team or an organisation has implemented something that works effectively, which will typically be a pragmatic mix of different delivery and management frameworks, it should then continuously evolve this to further

increase capability within the changing business environment. This concept is known as ‘Kaizen’ (continuous improvement) in Lean (Liker, 2004) (see Section 2.6.2).

MYTH: AGILE – JUST READ A BOOK

Understanding Agile is not something that can be achieved by simply reading a book. It is a very good idea to select some of the Agile books from the leading Agile exponents and read those; however, just reading a book cannot replace the practical experience that is essential to enabling an Agile mindset (see Section 2.1) and successfully transforming an organisation or team to become Agile.

This book serves as a fundamentals-level introduction to Agile, and we have referenced many other sources that we believe will help the reader on their Agile journey. However, as in most things, there is no substitute for practical experience.

MYTH: AGILE ONLY RELATES TO SOFTWARE DELIVERY

It is true that the Agile Manifesto describes Agile within the context of software delivery, but Agile can be applied successfully in business environments that are not exclusively software-related. In essence, Agile is suited to any dynamic business environment that experiences variability (e.g. marketing, business change and so on).

MYTH: AGILE SHOULD REPLACE EVERYTHING ALL AT ONCE (‘BIG BANG’ TRANSFORMATION)

When Agile is applied in a big-bang approach across large projects, programmes or across the entire organisation, there is a significant risk that the benefits of an Agile operating model (see Section 5.1) will not be realised or understood. Often the organisation and its staff will simply continue to do things as they have always done them while pretending – or believing – that they have moved to an Agile method.

Transforming capability is a long-term process of learning and change. Businesses evolve and the best way to do business evolves. It is therefore a mistake to implement a ‘big-bang’ Agile transformation and then just assume further improvement is no longer necessary.

Effective transformation is all about visualisation; a typical successful Agile transformation first transforms a vertical slice (i.e. a deliverable part of the overall system conceptualised as a ‘slice’ across the data, business rules and user interface (UI) layers of the architecture – see Section 13.2.1) of the entire target delivery organisation. It then measures and proves the effectiveness of the Agile approach before endeavouring to transform the remainder of the organisation. Without this tangible, factual, organisational visualisation it can be very difficult to get anyone to realistically transform.

In certain circumstances there may be a requirement for a big-bang transformation, for example, if an organisation’s competitors are already Agile, which may make competing with them very difficult or impossible. However, this is relatively rare

and needs high-quality support from experienced transformational Agile leads.

There is an interesting quote, which is generally attributed to Albert Einstein; it's referred to as Einstein's definition of insanity…

> *Insanity is doing the same thing over and over and expecting a different result.*

This statement nicely sums up failed Agile transformations where the organisation is still doing exactly what they have done previously (i.e. 'W'Agile – 'Waterfall pretending to be Agile'), and expecting added value even though they haven't actually changed anything.

MYTH: AGILE MEANS NO PLANNING – 'JUST DO IT'

The vast majority of Agile frameworks involve frequent, regular and evolutionary planning (see Section 7.3).

If a team is largely doing maintenance work or defect fixing (BAU work), or there is no need for the customer to look any further than a couple of weeks when creating a product, they will probably only be planning in single iterations/sprints.

If the customer does need to know roughly what product will be delivered within what timescale and at what cost, the team may be planning in releases containing iterations/sprints (one way to do this is by using 'Agile release trains' – see Section 14.8).

If there are 'release plans', a high-level agreement will be in place of what product is being delivered, and at what timescale and cost. However, the agreement and the plans are specifically set up to enable change, and there will be significant frequent, regular ongoing planning.

If the customer needs to get an understanding of the total baseline definition of a product within a baseline time and cost, then this could be delivered in multiple releases or an Agile project. If an Agile project is initiated, then high-level baseline definitions of the product and plans will be required; however, these are purposefully left at a high level as it is assumed that things will change as more is understood about the product.

Agile isn't 'just do it'; there is significant planning and re-planning involved when required.